

Forum:[Hack Forums](#)

Thread: [FastC4 - Cipher alg made in rust with pseudocode included](#)

derfdo716 - March 2, 2025, 10:30 am

This is a very simple cipher I designed in the rust language.

I wrote pseudocode for it as well which would make it very easy to translate into another language.

I would believe that security wise, this is similar to RC4 however it is at least 2x slower than RC4 as there are 2 iterations of the main cipher function called.

Like RC4 it also allows for an arbitrary sized key, in this cipher it must be at least 2 bytes of size.

```
Code pub struct FastC4 {
    state: Vec<u8>,
    len: usize
}

impl FastC4 {
    pub fn new(key: &Vec<u8>) -> FastC4 {
        if key.len() <= 1 {
            panic!("Key is too small");
        }

        FastC4 {
            state: key.clone(),
            len: key.len()
        }
    }

    fn routine(&mut self, data: &mut Vec<u8>) {
        let mut temp: u8 = 0x41;

        /*
            S = [...]
            D = [...]
            L = Len S
            T = 65
            for I = 0 to Len D - 1 {
                J = I mod L
                K = ~I mod L
                D[I] = D[I] ? S[J]
                S[J] = (S[J] + S[K] + T) ? 0xff
                T = T ? S[J]
                S[K] = (S[K] - S[J]) ? 0xff
            }
        */

        for i in 0..data.len() {
            let j = i % self.len;
            let k = (!i) % self.len;
            data[i] ^= self.state[j];
        }
    }
}
```

```

        self.state[j] = self.state[j].wrapping_add(self.state[k]).wrapping_add(temp);
        temp ^= self.state[j];
        self.state[k] = self.state[k].wrapping_sub(self.state[j]);
    }
}

fn apply_cipher(&mut self, data: &mut Vec<u8>) {
    let original_state = self.state.clone();

    self.routine(data);
    for i in 0..self.len { self.state[i] ^= original_state[(self.len - i) % self.len]; }
    self.routine(data);

    self.state = original_state; // Restore the state so that the decrypt operation can be performed
}

pub fn encrypt(&mut self, data: &mut Vec<u8>) { self.apply_cipher(data); } // The cipher encrypt
function is also the decrypt function
pub fn decrypt(&mut self, data: &mut Vec<u8>) { self.apply_cipher(data); }

}

```